New Generation Computing (2019) 37:67–96
https://doi.org/10.1007/s00354-018-0048-0

RESEARCH PAPER

Rule Extraction from Neural Network Using Input Data Ranges Recursively

Manomita Chakraborty[1] · Saroj Kumar Biswas[1] · Biswajit Purkayastha[1]

Received: 8 May 2018 / Accepted: 29 October 2018 / Published online: 11 November 2018

Abstract

Neural network is one of the best tools for data mining tasks due to its high accuracy. However, one of the drawbacks of neural network is its black box nature. This limitation makes neural network useless for many applications which require transparency in their decision-making process. Many algorithms have been proposed to overcome this drawback by extracting transparent rules from neural network, but still researchers are in search for algorithms that can generate more accurate and simple rules. Therefore, this paper proposes a rule extraction algorithm named Eclectic Rule Extraction from Neural Network Recursively (ERENNR), with the aim to generate simple and accurate rules. ERENNR algorithm extracts symbolic classification rules from a single-layer feed-forward neural network. The novelty of this algorithm lies in its procedure of analyzing the nodes of the network. It analyzes a hidden node based on data ranges of input attributes with respect to its output and analyzes an output node using logical combination of the outputs of hidden nodes with respect to output class. And finally it generates a rule set by proceeding in a backward direction starting from the output layer. For each rule in the set, it repeats the whole process of rule extraction if the rule satisfies certain criteria. The algorithm is validated with eleven benchmark datasets. Experimental results show that the generated rules are simple and accurate.

Keywords Neural network · Data mining · Rule extraction · Classification · Re-RX algorithm · RxREN algorithm

✉ Manomita Chakraborty
mou.look@gmail.com

Saroj Kumar Biswas
saroj@cse.nits.ac.in

Biswajit Purkayastha
biswajit@nits.ac.in

[1] Computer Science and Engineering Department, National Institute of Technology Silchar, Silchar, Assam 788010, India

Introduction

Over the last few decades, neural network (NN) has been an important area of research, especially for classification task due to its high accuracy on tremendous amount of highly nonlinear data [17]. Though it produces satisfactory accuracy, one of its main drawbacks is lack of transparency in decision-making process. That is an NN is unable to explain how it makes a final decision. Researchers tried to remove this drawback by extracting knowledge from NN in the form of human understandable rules like IF–THEN rules, M-of-N rules, oblique rules, and fuzzy rules [13, 16]. The development of various rule extraction algorithms enables NN to be suitable for those problems which require transparency in their decision-making process. Research in this area is still going on to generate more accurate, understandable and comprehensible rules.

Rule extraction process from NN follows three basic approaches: decompositional, pedagogical and eclectic [13, 16]. Decompositional approach is structure dependent which generates rule by analyzing hidden nodes and weight matrices of NN architecture. Pedagogical approach is a black box approach and generates rule as a whole in the form of input and output. Eclectic approach is a combination of both approaches.

Pedagogical and decompositional approaches both have advantages as well as disadvantages [2]. Pedagogical approach produces highly accurate rules but has exponential complexity, i.e., it is not effective when the size of NN increases. Pedagogical approach may not be able to capture all the valid rules. Whereas, decompositional approach is able to capture all the valid rules as it analyzes the structure of the network. But it is unsound, has unpredictable accuracy, and produces complex and larger rules. Compared to both of the approaches, eclectic approach is slower but effective and produces accurate rules as it combines the advantages of both approaches.

This paper proposes an eclectic rule extraction algorithm called Eclectic Rule Extraction from Neural Network Recursively (ERENNR) which analyzes each node and generates global rules. Many rule extraction algorithms based on eclectic approach have been proposed, but maximum of them have used magnitude of weights or decision tree while analyzing a node. Analyzing nodes based on magnitude of weights generates rule which are structure dependent and decision tree generates larger rule. Therefore, the proposed algorithm neither uses weights nor decision tree to generate rules.

The proposed ERENNR uses data ranges of input attributes to create a data range matrix for each hidden node and logical combinations of the outputs of hidden nodes to extract knowledge from the output nodes. Subsequently, by proceeding in the backward direction and using the extracted knowledge, the algorithm generates a set of rules in the form of input data ranges and outputs. The algorithm prunes each rule in the set if accuracy increases. For a rule in the set, the algorithm repeats the whole process recursively if the rule satisfies certain criteria.

The paper is organized as follows: Sect. 2 discusses the related works, Sect. 3 discusses the proposed algorithm in details, Sect. 4 illustrates the algorithm with an example, Sect. 5 presents experimental results with discussion, and finally Sect. 6 draws conclusion.

Related Works

Many rule extraction algorithms have been designed based on the three approaches which reveal the hidden information in NN in the form of symbolic rules. Though the algorithms extract rules based on the three approaches, the techniques used by the algorithms are different.

Algorithms like SUBSET and MofN [27] consider combination of weights for creating rules. SUBSET algorithm specifies an NN where the output of each neuron in the network is either close to zero or close to one. The algorithm finally searches for subsets of incoming weights that exceed the bias on a unit. MofN algorithm is an extension to the SUBSET algorithm, which clusters the weights of a trained network into equivalent classes and extracts m-of-n style rules.

Various algorithms like NeuroRule [21], Greedy Rule Generation (GRG) [18], and Rule Extraction (RX) [22] deal with discretized inputs. NeuroRule generates each rule by an automatic rule generation method which covers as many samples from the same class as possible with the minimum number of attributes in the rule condition. Automatic rule generation method generates rules that explain the network's output in terms of the discretized hidden unit activation values and discretized activation values in terms of the discretized attributes of the input data. RX algorithm recursively generates rules by analyzing the discretized hidden unit activations of a pruned network with one hidden layer. When the number of input connections to a hidden unit is larger than a certain threshold, a new NN is created and trained with the discretized activation values as the target outputs. Otherwise, rules are obtained that explain the hidden unit activation values in terms of the inputs. GRG uses greedy technique to generate rules with discrete attributes, i.e., at each iteration it searches for the best rule.

Algorithms like NeuroLinear [23] and BRAINNE [20] do not require discretization of inputs. NeuroLinear extracts oblique rules from neural network with continuous attributes. BRAINNE extracts global rule from neural network without discretization of input attributes.

Binarized Input–Output Rule Extraction (BIO-RE) algorithm [26] works only with binary input whereas Orthogonal Search based Rule Extraction algorithm (OSRE) [8] can be applied to data with nominal or ordinal attributes. BIO-RE simplifies the representation of underlying logic of a trained neural network using k-map, algebraic manipulations or a tabulation method. OSRE algorithm converts given input to 1 from N form and then performs rule extraction based on activation responses.

Full-RE [26], Rule extraction by Reverse Engineering the Neural Network (RxREN) [3], and Rule Extraction from Neural Network using Classified and Misclassified data (RxNCM) [5] algorithms can work with any type of attributes. Full-RE algorithm extracts rule with certainty factor from feed forward neural network that is trained by any type of attributes. RxREN generates rules in the form of data ranges of inputs from mixed dataset and relies on the reverse engineering technique to prune the insignificant input neurons. RxNCM is an extension to RxREN algorithm. RxREN uses only misclassified data, whereas RxNCM uses both classified and misclassified data to find the data ranges of significant attributes.

Few algorithms like Trepan [7], Artificial Neural Network Tree (ANNT) [1], Recursive Rule Extraction (Re-RX) [25], Ensemble-Recursive-Rule extraction (E-Re-RX) [14], Reverse Engineering Recursive Rule Extraction (RE-Re-RX) [6], Continuous Re-RX [10, 12], the Re-RX algorithm with J48graft [12], Sampling Re-RX [12], Sampling Re-RX with J48graft [11], Fast Extraction of Rules from Neural Networks (FERNN) [24], and Hierarchical and Eclectic Rule Extraction via Tree Induction and Combination (HERETIC) [15] use decision tree as a part of the rule extraction process. TREPAN algorithm extracts a decision tree from a trained network which is used as an "oracle" to answer queries during the learning process. ANNT algorithm maps each layer using decision tree to generate rules from pruned network. Re-RX algorithm uses decision tree to generate rules for discrete attributes and generates separate rules for discrete and continuous attributes using a recursive process. E-Re-RX, RE-Re-RX, continuous Re-RX, Sampling Re-RX, Re-RX with J48graft, and Sampling Re-RX with J48graft all come under the family of Re-RX, i.e., all uses decision tree as a part of their rule extraction process. E-Re-RX algorithm is an ensemble version of Re-RX algorithm which generates primary rules followed by secondary rules and finally these rules are integrated to obtain the final set. RE-Re-RX extends Re-RX by replacing the linear hyperplane for continuous attributes with simpler rules in the form of input data ranges. Continuous Re-RX uses C4.5 decision tree to generate rules for both discrete and continuous attributes using recursive approach. Re-RX with J48graft replaces the conventional Re-RX algorithm, which uses C4.5 as a decision tree with J48graft. Sampling Re-RX uses sampling techniques for preprocessing with an objective to generate concise and accurate rules. Sampling Re-RX with J48graft algorithm uses both sampling and J48graft. FERNN is another such algorithm which identifies the relevant hidden units based on decision tree and finds the set of relevant network connections between input and hidden units based on magnitudes of their weights. HERETIC uses decision tree on each node to generate rules.

Active Learning-based Pedagogical Approach (ALPA) [9] algorithm is also proposed which can extract rules from any black box. ALPA generates rule by generating new artificial data points around training vectors with low confidence score.

ERENNR Algorithm

ERENNR algorithm generates classification rules in the form of input data ranges and targets. The algorithm extracts rules by analyzing each node of the pruned network. It uses data ranges of input attributes and logical combination of hidden outputs to analyze hidden and output nodes, respectively. Subsequently, it combines knowledge obtained from each node to construct rule set and prunes each rule in the set if accuracy increases. For each rule in the set, it repeats the whole process of rule extraction if the rule satisfies certain conditions. The flow chart for the algorithm is given below in Fig. 1.

The details of the algorithm are given below:

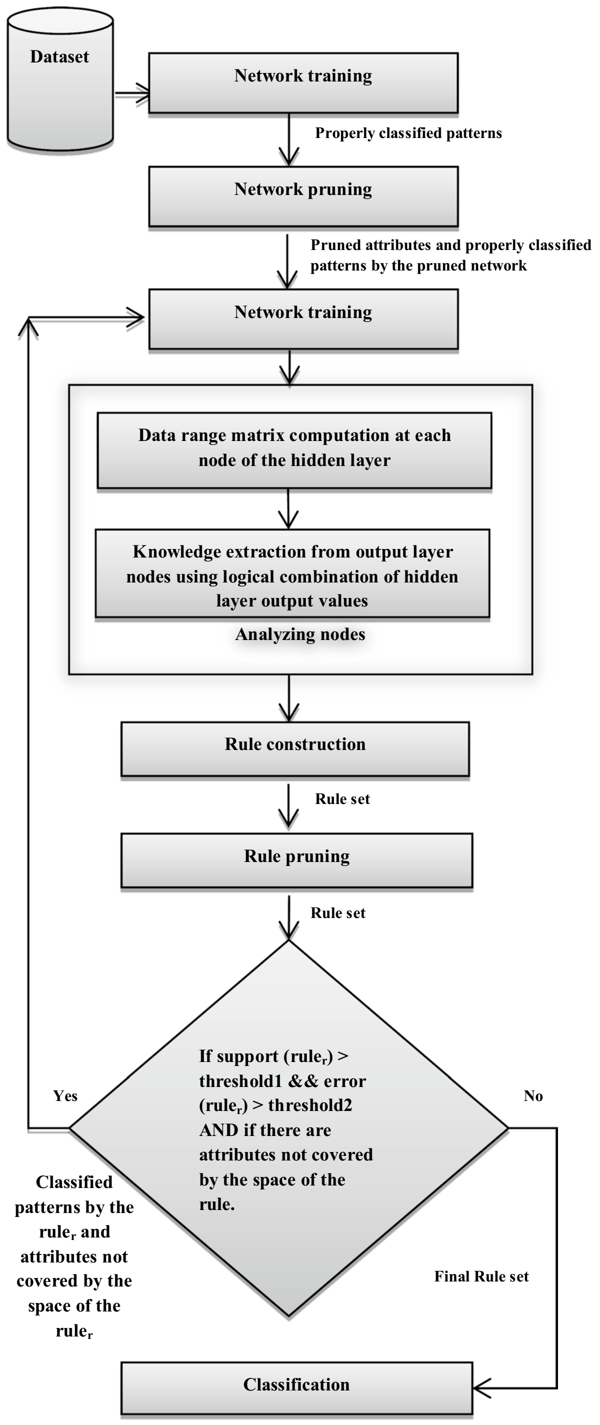
Fig. 1 Flow chart of ERENNR

Dataset
Network training
Properly classified patterns
Network pruning
Pruned attributes and properly classified patterns by the pruned network
Network training
Data range matrix computation at each node of the hidden layer
Knowledge extraction from output layer nodes using logical combination of hidden layer output values
Analyzing nodes
Rule construction
Rule set
Rule pruning
Rule set
If support (rule_r) > threshold1 && error (rule_r) > threshold2 AND if there are attributes not covered by the space of the rule.
Yes
No
Classified patterns by the rule_r and attributes not covered by the space of the rule_r
Final Rule set
Classification

Network Training

The algorithm uses a feed forward neural network (FFNN) with one hidden layer and back-propagation (BP) algorithm for training. The number of hidden nodes is calculated by varying the number from (l + 1) to 2 l where l is the number of input attributes. The network architecture which gives the smallest mean square error is selected as the optimal architecture [4, 19] and this architecture is taken for further experimentation.

Network Pruning

The algorithm uses the pruning concept used in RxREN [3]. For each input of the trained NN, the algorithm finds the number of incorrectly classified patterns. It finds a threshold value equal to the least value among all the numbers of incorrectly classified patterns and then removes those input(s) which have number of misclassified patterns equal to the threshold to form a temporary pruned network. The algorithm considers this temporary pruned network as the pruned network if classification accuracy of the network increases on training set. It repeats the whole process of pruning till the classification accuracy of the pruned network increases.

The network pruning algorithm is given below.

Notations:
N: Denotes FFNN.
D: Denotes training dataset.
T: Represents a set of correctly classified examples by the trained ***N*** on ***D***.
$\boldsymbol{N_{acc}}$: Denotes accuracy of trained ***N*** on ***D***.
I: Represents a set of input neurons of ***N***.
$\boldsymbol{I_i}$: Denotes $\boldsymbol{i^{th}}$ input neuron in the set ***I***.
d : Denotes number of inputs in set ***I***.
$\boldsymbol{E_i}$: Represents set of incorrectly classified examples by training ***N*** in absence of $\boldsymbol{I_i}$.
$\boldsymbol{err_i}$: Denotes the number of incorrectly classified examples in $\boldsymbol{E_i}$.
B : Represents a set of insignificant input neurons.
TPN: Denotes a temporary pruned network.
PN: Denotes a pruned network.
$\boldsymbol{P_{acc}}$: Denotes accuracy of trained ***TPN*** on ***D***.
$\boldsymbol{I'}$: Represents a set of input neurons of ***TPN***.
T': Represents a set of correctly classified examples by trained ***TPN*** on ***D***.

Input: A trained FFNN.
Output: A pruned network.

```
||Network pruning(N, T, I, $N_{acc}$)||

Step1: For each input neuron $I_i$ , find $E_i$ on set T.
Step2: Compute the threshold $\theta = \min(err_i), i=1,..,d$.
Step3: Frame the set $B = \{I_i \mid err_i = \theta\}$.
Step4: Form the TPN by removing all the insignificant input neurons of B.
Step5: Compute $P_{acc}$.
Step6: If ($P_{acc} \geq N_{acc}$), then PN=TPN,
        • Assign N=PN, $N_{acc} = P_{acc}$ , T= T', I= I' .
        • Call Network pruning (N, T, I, $N_{acc}$) process.
       Else stop the process.
```

Recursive Rule Extraction

Network is again trained with the properly classified patterns and the attributes of the pruned network. Training is done here to restrict the output activation values of hidden nodes to + 1 and − 1. This is done using a bipolar sigmoidal activation function with large gain factor β, greater than 100. Large gain factor converts the hidden output with small value to − 1 and large value to + 1. The formula of the activation function is given below in (1):

$$\delta(x) = \frac{1 - e^{-\beta x}}{1 + e^{-\beta x}} \tag{1}$$

Rule extraction process is done by the following three steps.

Analyzing Nodes:

The algorithm extracts a data range matrix from each hidden node. Data range matrix is a table where each row represents an input attribute and each column represents an output value of a hidden node. ERENNR finds the classified patterns for each hidden output and computes minimum and maximum value of the classified patterns for each attribute.

Figure 2 shows a data range matrix that is extracted from a hidden node. a_1, a_2,…, a_n are the attributes of pruned network. 1 and -1 are activation values of the hidden node. Elements of the matrix show the ranges of attributes in respective activation output. L_{ij} denotes lower range and U_{ij} denotes upper range of an attribute a_i in activation output j. (i ranges from 1 to n and $j=\{1, -1\}$).

Thereafter creating data range matrix for each hidden node, ERENNR extracts knowledge from the output layer. The algorithm finds the logical combination of hidden outputs for each of the properly classified patterns with respect to each class of the output node. Among all the combinations in each class, ERENNR finds the unique combinations by removing the redundant ones. There may be some combinations which classify patterns in all the classes. Those combinations are useless. So, the algorithm removes those combinations to generate the final set of combinations or temporary rules in each class. The example given below shows final set of logical combinations of hidden outputs with respect to classes where k is the number of hidden nodes and m is the number of classes in the output layer.

If (hidden_1= 1 and hidden_2= -1 and hidden_3= 1 and … hidden_k= -1), then class=1.

If (hidden_1= 1 and hidden_2= 1 and hidden_3= -1 and … hidden_k= -1), then class=1.

If (hidden_1= -1 and hidden_2= -1 and hidden_3= -1 and … hidden_k= -1), then class=2.

:

:

If (hidden_1= -1 and hidden_2= -1 and hidden_3= 1 and … hidden_k= 1), then class=m.

Fig. 2 Data range matrix of a hidden node

$$\begin{array}{c|cc} & 1 & -1 \\ \hline a_1 & [L_{11}\text{-}U_{11}] & [L_{1(-1)}\text{-}U_{1(-1)}] \\ a_2 & [L_{21}\text{-}U_{21}] & [L_{2(-1)}\text{-}U_{2(-1)}] \\ . & . & . \\ . & . & . \\ a_n & [L_{n1}\text{-}U_{n1}] & [L_{n(-1)}\text{-}U_{n(-1)}] \end{array}$$

Rule Construction:

Proceeding in a backward direction starting from the output layer, ERENNR generates rules in the form of data ranges of input and output classes. For each temporary rule of output layer, ERENNR selects data ranges of all attributes for each hidden output value in the rule using the knowledge that is extracted in the preceding layer. For example, consider the temporary rule given below:

If ($hidden_1$= 1 and $hidden_2$= -1 and $hidden_3$= 1 and… $hidden_k$= -1), then class=1.

Firstly, ERENNR generates data ranges of all attributes for $hidden_1 = 1$, $hidden_2 = -1$, $hidden_3 = 1$,…, and, $hidden_k = -1$ separately using the data range matrices of each of the hidden nodes. Table 1 shows data ranges of all attributes for each hidden output in the temporary rule.

Then, ERENNR combines all the hidden outputs in a temporary rule by finding the minimum lower range and maximum upper range for each attribute. For example in case of attribute a_1 in Table 1 among all the hidden nodes from 1 to k, the algorithm finds the minimum L_{1j} and maximum U_{1j} to select the data range for attribute a_1. The rule generated after combination is shown below where L_i and U_i are lower and upper range of attribute $a_{i,}$ respectively.

If (L_1<=a_1<= U_1 and L_2<=a_2<= U_2 and… and L_n<=a_n<= U_n), then class=1.

Similarly, for all the temporary rules of the output layer, rules are generated. The rule construction step still may generate some redundant and useless rules and consequently ERENNR removes those rules. An example of rule set is given below:

Rule 1: If ($L1_1$<=a_1<= $U1_1$ and $L1_2$<=a_2<= $U1_2$ and… and $L1_n$<=a_n<= $U1_n$), then class=1.

Rule 2: If ($L2_1$<=a_1<= $U2_1$ and $L2_2$<=a_2<= $U2_2$ and… and $L2_n$<=a_n<= $U2_n$), then class=1.

Rule 3: If ($L3_1$<=a_1<= $U3_1$ and $L3_2$<=a_2<= $U3_2$ and… and $L3_n$<=a_n<= $U3_n$), then class=2.

.
:
:

Rule r: If (Lr_1<=a_1<= Ur_1 and Lr_2<=a_2<= Ur_2 and… and Lr_n<=a_n<= Ur_n), then class=m.

Table 1 Data ranges of all attributes for each hidden output in a temporary rule

Hidden	a_1	a_2	…	a_n
$hidden_1 = 1$	[L_{11}–U_{11}]	[L_{21}–U_{21}]	…	[L_{n1}–U_{n1}]
$hidden_2 = -1$	[$L_{1(-1)}$–$U_{1(-1)}$]	[$L_{2(-1)}$–$U_{2(-1)}$]	…	[$L_{n(-1)}$–$U_{n(-1)}$]
$hidden_3 = 1$	[L_{11}–U_{11}]	[L_{21}–U_{21}]	…	[L_{n1}–U_{n1}]
:	:	:	:	:
$hidden_k = -1$	[$L_{1(-1)}$–$U_{1(-1)}$]	[$L_{2(-1)}$–$U_{2(-1)}$]	…	[$L_{n(-1)}$–$U_{n(-1)}$]

L_{1i} and U_{1i} denote lower range and upper range of attribute a_i in rule l, respectively.

Rule Pruning:

Firstly, to make the range more appropriate, the algorithm adjusts the range of each attribute in a rule. For each condition in a rule, it removes the equal sign and checks whether accuracy increases or not. If accuracy increases, then the equal sign is removed from the condition. Shifting of range helps to determine more specific range, if the ranges of attributes overlap for different classes.

Thereafter, rule pruning is performed to make the rules simple and accurate. Pruning is done by removing each condition from a rule. If the accuracy on properly classified patterns by the pruned network increases after removing a condition, the condition is removed from the rule. After rule pruning, the rule which covers the maximum number of attributes is given the first preference in the rule set.

The algorithm evaluates each rule in the rule set in terms of error and support. The support of a rule is the percentage of classified patterns by the rule and the error is the percentage of misclassified patterns by the rule. If the error exceeds a specified threshold and the support meets the minimum threshold, then ERENNR further divides the subspace of the rule by calling the recursive rule extraction step of the algorithm. The step proceeds with those attributes of the pruned network which are not present in the subspace of the rule and with the patterns classified by the rule.

The recursive rule extraction algorithm is given below:

Notations:
P: Denotes pruned network
$\boldsymbol{a}$: Represents a set of attributes of ***P***.
$\boldsymbol{a}_i$:Denotes i^{th} attribute of set ***a***.
C : Represents a set of correctly classified patterns by ***P***.
$\boldsymbol{H}_h$: Denotes h^{th} hidden node of the hidden layer ***H***.
$\boldsymbol{DM}_h$: Denotes data range matrix of hidden node ***h.***
$\boldsymbol{O}_t$: Denotes t^{th} output of a hidden node.
$\boldsymbol{c}_k$: Denotes k^{th} class of output layer.
TR: Denotes temporary rule set of output layer.
$\boldsymbol{TR}_{tr}$: Denotes $(tr)^{th}$ rule of ***TR.***
L: Denotes lower range.
U: Denotes upper range.
R: Denotes rule set.
$\boldsymbol{R}_r$: Denotes r^{th} rule of ***R.***
$\boldsymbol{Acc\_R}_r$: Denotes accuracy of $\boldsymbol{R}_r$.
a': Represents a set of attribute(s) not covered by subspace of $\boldsymbol{R}_r$ but present in the set ***a.***
C': Represents a set of classified patterns by $\boldsymbol{R}_r$ on ***C.***

Input: A pruned network.
Output: A set of rules.

```
|| Recursive rule extraction(P, C, a) ||

Step1: Train $P$ with $a$ and $C$ using back propagation with nonlinear activation function having large gain factor.
Step2: At each $H_h$ :
        ➢ Find the $L$ and $U$ of the patterns belonging to $C$ for each attribute $a_i$ with respect to $O_t$ of $h$ to create $DM_h$.
Step3: Find logical combination of $H_h=O_t$ for each of the patterns in $C$ with respect to $c_k$. Remove the redundant and useless logical combinations to form $TR$.
Step4: Construct $R$ by superimposing each $TR_{tr}$ with $DM_h$ of each $H_h$ present within each $TR_{tr}$.
    ➢ Construct $R_r$:
        ➢ Replace each hidden value $H_h= O_t$ in $TR_{tr}$ with data ranges of all attributes $a_i$ in $DM_h$ for $O_t$.
        ➢ Construct $R_r$ by finding the min($L$) and max($U$) of each $a_i$ for all $H_h$ in $TR_{tr}$
Remove $R_r$ from $R$ if it is redundant or useless.
Step5:
    ➢ For each $R_r$ in $R$,
        ➢ For each condition in $R_r$ , remove equal to sign and check $Acc\_R_r$ on $C$
            ▪ If $Acc\_R_r$ increases, the equal to sign is removed.
        ➢ For each condition in $R_r$, remove the condition and check $Acc\_R_r$ on $C$.
            ▪ If $Acc\_R_r$ increases, the condition is removed.
    ➢ Arrange all $R_r$ in $R$ in descending order according to number of attributes covered by each $R_r$.
Step6: For each $R_r$ in $R$,
    ➢ If support($R_r$) > threshold1 and error($R_r$) > threshold2
        ➢ If $a'$ is present then call Recursive rule extraction (P, C', a').
        ➢ Else stop.
    ➢ Else stop.
```

Table 2 Accuracy of NN and pruned NN for Australian Credit Approval dataset

Training accuracy of NN (%)	Testing accuracy of trained NN (%)	Training accuracy of pruned NN (%)	Testing accuracy of pruned NN (%)
87.99	84.54	87.99	85.02

Illustrative Example

To illustrate the algorithm, Australian Credit Approval dataset is taken from UCI repository. Australian Credit Approval is a standard dataset used for comparison of different rule extraction algorithms. There is a good mix of attributes with 690 patterns and two classes in the dataset. It contains 6 continuous attributes, 8 categorical attributes, and 1 binary attribute for class. 70% of the patterns (483 patterns) are taken as training set and 30% of the patterns (207 patterns) as testing set. A single hidden layer FFNN with 25 hidden nodes is trained using back-propagation algorithm. After pruning, attribute 2 is removed from the network. Table 2 shows the accuracy of NN and pruned NN.

425 patterns are correctly classified by the pruned NN. The pruned network is again trained by the 425 patterns. Data range matrix for each hidden node is created. Table 3 shows the data range matrix of hidden node 1.

Logical combinations of hidden outputs are formed for each of the 425 patterns. After removing the useless and redundant combinations, 124 important combinations are selected to form temporary rule set. For each of the 124 temporary rules, final rules are created using the data ranges of attributes for each hidden output value present in the temporary rule. For example, one of the temporary rules for the dataset is given below:

If (hidden$_1$= -1 ^ hidden$_2$=-1 ^.... hidden$_7$=1 ^ hidden$_8$=1 ^....
^ hidden$_{25}$=-1), then class=0.

Table 4 shows the data ranges of all attributes for each hidden output in the temporary rule.

Rule is created using Table 4 by selecting minimum lower and maximum upper range of each attribute. The generated rule is given below.

If ((0<= attribute1 <=1) AND (0<= attribute3 <=28) AND...
AND (1<= attribute14 <=50001)), then class=0.

Similarly, rules are generated for all the temporary rules in the set. After removing the useless and redundant rules, the rule set contains 2 rules. The pruned rule set after rule pruning is given below.

Rule 1: If (attribute10 >0), then class=1.
Rule 2: If (attribute8<=1), then class=0.
Default rule: Else class 1.

Table 3 Data range matrix of hidden node1 for Australian Credit Approval dataset

Attribute	+ 1	− 1
1	[0–1]	[0–1]
3	[0–28]	[0–25.1250]
4	[1–3]	[1, 2]
5	[1–14]	[1–14]
6	[1–9]	[1–9]
7	[0–28.5]	[0–18]
8	[0–1]	[0–1]
9	[0–1]	[0–1]
10	[0–40]	[0–23]
11	[0–1]	[0–1]
12	[1–3]	[1, 2]
13	[0–2000]	[0–640]
14	[1–205]	[1–50,0001]

Table 4 Data ranges of attributes for Australian Credit Approval dataset

Hidden	Attribute 1	Attribute 3	…	Attribute 14
$\text{hidden}_1 = -1$	[0–1]	[0–25.1250]	….	[1–50001]
$\text{hidden}_2 = -1$	[0–1]	[0–28]	…	[1–16]
…	…	…	…	…
$\text{hidden}_{25} = -1$	[0–1]	[0–25.1250]	….	[1–50,001]

Table 5 Support and error of Rule 1 and Rule 2 for Australian Credit Approval dataset

Rules	Classified instances	Correctly classified	misclassified	Support	Error
Rule 1	183	138	45	0.4306	0.2459
Rule 2	242	188	54	0.5694	0.2231
Default rule	0	0	0	–	–

Subspace of Rule 1 and Rule 2 is further subdivided as support and error for both the rules that are greater than the specified thresholds. The thresholds for support and error in this experiment are set to 0.05. Table 5 shows that the support and error for both the rules are greater than 0.05. Consequently, Recursive rule extraction (P, C, a) step is called for both the rules. For Rule 1, the step is repeated with $C = 183$ patterns and $a = \{a\text{-attribute10}\}$. For Rule 2, the step is repeated with $C = 242$ patterns and $a = \{\text{a-attribute8}\}$. The process continues till the support and error for each generated rule are greater than the thresholds.

The final set of rules is given below:

```
Rule 1: If (attribute10 >0), then follows:
    Rule 1.1: If (attribute8==0), then class=0.
    Rule 1.2: Else class=1.

Rule 2: If (attribute 8<=1), then follows:
    Rule 2.1: If ((attribute 6<=9) ^ (attribute 13<=980)),
              then class=0.
    Rule 2.2: If ((attribute 9>=0) ^ (attribute 11<=1)), then
              class=1.
    Rule 2.3: Else class=0.
Default rule: Else class 1.
```

Accuracies of final rule set on training and testing sets are 85.92% and 85.99%, respectively. The threshold values of support and error are important for determining accuracy as well as complexity [25]. The values are selected experimentally. Rule sets are generated by varying the threshold values, and the rule set which gives best performance is considered. Experimental results with various values of threshold for Australian Credit Approval dataset are shown in Table 6. Best testing accuracy is obtained when support and error are equal to 0.05. With large thresholds, fewer rules are obtained and with small thresholds, many rules are obtained.

Experimental Results

Data sets and Experimental Set Up

All the datasets used for the experiments are listed in Table 7. They are collected from UCI and Keel machine learning repository. Among all the datasets, seven are mixed dataset and four are continuous dataset. Experiments on all the datasets are done by MATLAB in windows environment. Experiments are done for 70–30 partition (70% patterns of a dataset as training set, and 30% patterns as test set) and for 80–20 partition (80% patterns of a dataset as training set, and 20% patterns as test set). 5-fold and tenfolds cross-validation are also performed. A single hidden layer FFNN is trained with back-propagation algorithm. For each network architecture, Mean Square Error (MSE) is measured on training set and the architecture which finds minimum MSE is selected for further experimentation and is called optimal architecture of the network [4, 19].

Table 6 Accuracy and number of rules with different support and error for Australian Credit Approval dataset

Support, error	Training accuracy	Testing accuracy	Number of rules
0.1, 0.1	77.02	76.81	3
0.05, 0.05	85.92	85.99	8
0.05, 0.02	86.33	83.12	12
0.05, 0	85.92	83.54	13

Table 7 Datasets used for experiments

Dataset	No. of patterns	Attribute (including class attribute)	Attribute characteristics	No. of classes
Credit approval	690	16	Categorical, integer, real	2
Australian credit approval	690	15	Categorical, integer, real	2
Echocardiogram	132	13	Categorical, integer, real	2
Statlog (heart)	270	14	Categorical, real	2
Breast cancer	699	11	Real	2
Blood transfusion	748	5	Real	2
German	1000	21	Categorical, integer	2
Eye	14,980	15	Integer, real	2
Pima Indians diabetes	768	9	Integer, real	2
Census income	48,842	15	Categorical, integer	2
Thyroid	7200	22	Continuous, binary	3

Rule Extraction by ERENNR

The set of rules generated for Australian Credit Approval dataset is shown in illustrative example section. Rules generated with rest of the datasets for 70–30 partition are shown below.

Credit Approval:

After removing patterns with missing values, 458 patterns are selected for training and 196 patterns are selected for testing. The number of hidden nodes is 26. The set of rules extracted by ERENNR is shown below:

Rule 1: If ((attribute 4 <=2) ^ (attribute 9>=2)), then class=0.
Default rule: Else class=1.

Echocardiogram:

There are total 132 patterns among which 71 patterns are removed as they contain missing values. So, 43 patterns are selected for training and 18 patterns are selected for testing. The number of hidden nodes is 17. The set of rules extracted by ERENNR is shown below:

```
Rule 1: If ((attribute1>0.5) ^ (attribute1<=57) ^ (attribute
        2==0)), then class=0.
Rule 2: If (attribute2==1), then follows:
    Rule 2.1: If (attribute1<22), then class=1.
    Rule 2.2: Else class=0.
Default rule: Else class=0.
```

Statlog (Heart):

189 patterns are selected for training and 81 patterns are selected for testing. The number of hidden nodes is 22. The set of rules extracted by ERENNR is shown below:

```
Rule 1: If ((attribute13>3) ^ (attribute13<=7)), then
        follows:
    Rule 1.1: If ((attribute3 <4) ^ (attribute8<195) ^
              (attribute12<3)), then follows:
        Rule 1.1.1: If ((attribute6>0) ^ (attribute7>1)),
                    then class=0.
        Rule 1.1.2: If ((attribute1 < 70) ^ (attribute4 <
                    180)), then class=1.
        Rule 1.13: Else class=0.
    Rule 1.2: If (attribute4>94), then class=0.
    Rule 1.3: Else class=1.
Rule 2: If (attribute12<3), then follows:
    Rule 2.1: If (attribute13 >=3), then class=1.
    Rule 2.2: Else class=0.
Default rule: Else class=0.
```

Breast Cancer:

After removing the useless attribute (first attribute) and patterns with missing values, it contains 683 patterns and 9 attributes. 479 patterns are selected for training and 204 patterns are selected for testing. The number of hidden nodes is 11. The set of rules extracted by ERENNR is shown below:

```
Rule 1: If ((attribute1<=6) ^ (attribute2<=4) ^ (attribute3 <=
        4) ^ (attribute8<=3)), then class=0.
Default rule: Else class=1.
```

Blood Transfusion:

The dataset contains 4 attributes with 748 patterns. 523 patterns are selected for training and 225 patterns are selected for testing. The number of hidden nodes is 8. The set of rules extracted by ERENNR is shown below:

```
Rule 1: If (attribute2< 46), then class=0.
Default rule: Else class=1.
```

German:

700 patterns are selected for training and 300 patterns are selected for testing. The number of hidden nodes is 27. The set of rules extracted by ERENNR is shown below:

Rule 1: If ((attribute1>1) ^ (attribute5 <15945)), then follows:
  Rule 1.1: If ((attribute2>=4 ^ attribute2 <=60) ^ (attribute3 >=1 ^attribute3 <=5) ^ (attribute4 >=1 ^ attribute4 <=11)), then class=1.
  Rule 1.2: Else class=0.
Default rule: Else class=0;

Eye:

10486 patterns are selected for training and 4494 patterns are selected for testing. The number of hidden nodes is 25. The set of rules extracted by ERENNR is shown below:

Rule 1: If ((attribute1>=4197.95) ^ (attribute1<309231) ^ (attribute10>=1816 .41) ^ (attribute10<4323.08)), then follows:
  Rule 1.1: If ((attribute13 >= 3504.1) ^ (attribute13 < 4757.95)), then class=0.
  Rule 1.2: Else class=1.
Default rule: Else class=1.

Pima Indian Diabetes:

538 patterns are selected for training and 230 patterns are selected for testing. The number of hidden nodes is 14. The set of rules extracted by ERENNR is shown below:

Rule 1: If ((attribute1 <11) ^ (attribute2<155) ^ (attribute3>0) ^ (attribute6<50) ^ (attribute8<70)), then follows:
  Rule 1.1: If ((attribute5<600) ^ (attribute7<2.288)), then class=0.
  Rule 1.2: Else class=1.
Rule 2: If (attribute2>106), then class=1.
Default rule: Else class=0.

Census Income:

After removing patterns with missing values, 31,655 patterns are selected for training and 13,567 patterns are selected for testing. The number of hidden nodes is 26. The set of rules extracted by ERENNR is shown below:

Rule 1: If ((attribute11 >0) ^ (attribute12>0)), then class=1.
Rule 2: If (attribute11<99999), then class=0.
Default rule: Else class=1.

Thyroid:

It is a highly imbalanced dataset with three classes: normally functioning (class 3), under-functioning (primary hypothyroidism, class 2), or overactive (hyperthyroidism, class 1). Class 1 and class 2 represent 2.3% (166 patterns) and 5.1% (367 patterns) of the dataset, respectively, and the remaining 92.6% (6667 patterns) are classified as class 3. It contains 6 continuous and 15 binary attributes. 5040 patterns are taken as training set, 2160 patterns are taken as testing set, and the number of hidden nodes used is 30. The set of rules extracted by the ERENNR algorithm is shown below:

Rule 1: If ((atribute3==0) ^ (attribute8==0) ^ (attribute17>=0.0065) ^ (attribute19<=0.15) ^ (attribute20>=0.0560) ^ (attribute21)>=0.0649)), then class=2.
Rule 2: If ((attribute17<0.1770) ^ (attribute19>=0.019)), then class=3.
Default rule: Else class=1.

The testing accuracies of NN and the above rule sets for all the datasets are shown in Table 8.

Performance and Comparison of ERENNR with Existing Methods

Till now many rule extraction algorithms have been proposed. Here, comparison is shown with RxREN [3] and Re-RX [25] as ERENNR extracts rule in the form of data ranges of input like RxREN and extracts rule recursively like Re-RX. Moreover, Re-RX and RxREN have already been compared with many well-known rule extraction algorithms like NeuroRule [21], NeuroLinear [23], GRG [18], etc.

Table 9 shows the comparison of ERENNR with Re-RX and RxREN for 80–20 partition and 70–30 partition, respectively, in terms of accuracy. For all the

Table 8 Accuracy of ERENNR on testing sets for 70–30 fold

Datasets	NN (%)	Rules (threshold1 = 0.05 && threshold2 = 0.05) (%)
Credit approval dataset	83.67	85.20
Australian credit approval	84.54	85.99
Echocardiogram	94.44	100
Statlog (heart) dataset	70.37	77.78
Breast cancer	98.43	97.06
Blood transfusion	76.44	75.56
German	77	73
Eye	65.82	65.55
Pima Indians diabetes	76.09	79.13
Census income	77.98	76.72
Thyroid	90.51	95.31

Table 9 Comparison of testing accuracy (in %)

Datasets	80–20 partition			70–30 partition		
	Re-RX	RxREN	ERENNR	Re-RX	RxREN	ERENNR
Credit approval dataset	91.60	93.13	93.89	81.12	81.12	85.20
Australian credit approval	72.46	76.81	81.88	75.36	84.54	85.99
Echocardiogram	91.67	91.67	100	94.44	94.44	100
Statlog (heart) dataset	75.93	70.37	77.78	67.90	72.83	77.78
Breast cancer	91.91	94.12	97.79	93.13	95.59	97.06
Blood transfusion	78.67	78.67	78.67	74.67	75.56	75.56
German	74	71.5	75.5	71	67	73
Eye	68.66	67.62	69.06	55.03	65.26	65.55
Pima Indians diabetes	74.68	74.03	83.12	69.13	71.30	79.13
Census income	69.94	75.79	76.37	70.75	76.18	76.72
Thyroid	92.01	93.06	94.24	93.95	94.94	95.31

The performance is also evaluated by mean across 5-folds and 10-folds. Table 10 shows the results for 5-fold cross-validation and tenfolds cross-validation. It is observed that the proposed ERENNR algorithm performed better than Re-RX and RxREN for all the datasets when different folds are taken into consideration.

Like accuracy, fidelity is one more criteria to evaluate the performance of extracted rules which is the ability of the rules to mimic the behavior of NN. Fidelity of rule set R is shown by (2). T denotes test set, and $f_{\mathrm{NN}}(t)$ and $f_{\mathrm{R}}(t)$ denote the function implemented by a trained NN and the rules extracted from the trained NN, respectively [28]. Table 11 shows the comparison of fidelity for 80–20 partition and 70–30 partition, respectively, and Table 12 shows the comparison of mean fidelity for 5-folds and tenfolds, respectively. All the results show that fidelity of rules by ERENNR is higher than Re-RX and RxREN in maximum of the datasets.

$$\mathrm{Fidelity}_{\mathrm{R}} = 1 - \mathrm{probability}\{t \in T | f_{\mathrm{R}}(t) \neq f_{\mathrm{NN}}(t)\} \tag{2}$$

Only accuracy is not sufficient to validate the performance of a classification model. This paper considers other measures like recall, FP rate, specificity, precision, f-measure, and MCC to validate the performance. These performance measures are calculated using confusion matrix. Confusion matrix contains information about actual classification and predicted classification done by a classifier. Table 13 shows the confusion matrix for a binary classifier with the following data entries: (a) True positive (TP) is the number of 'positive' instances categorized as 'positive', (b) False positive (FP) is the number of 'negative' instances categorized as 'positive', (c) False negative (FN) is the number of 'positive' instances categorized as 'negative' and (d) True negative (TN) is the number of 'negative' instances categorized as 'negative'. Performance measures defined based on the 2 class matrix are shown in Table 14.

Tables 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, and 25 show the comparison of ERENNR with Re-RX and RxREN based on the performance measures. A good

Table 10 Comparison of testing accuracy with cross-validation results

Datasets	5-fold cross-validation (in %)			tenfold cross-validation (in %)		
	Re-RX	RxREN	ERENNR	Re-RX	RxREN	ERENNR
Credit approval dataset	79.40	84.43	86.11	81.39	84.46	86.62
Australian credit approval	70.89	73.04	85.36	73.2	73.76	85.51
Echocardiogram	90	93.33	98.33	93.33	93.33	96.67
Statlog (heart) dataset	72.59	73.33	74.81	73.33	75.93	80.37
Breast cancer	90.59	91.47	95.58	90.29	93.09	96.47
Blood transfusion	76.26	76.40	76.53	73.6	76.39	76.53
German	71	67.4	72.20	70.4	67.4	73.1
Eye	53.98	54.95	55.34	53.98	54.65	55.24
Pima Indians diabetes	67.27	70.26	75.84	68.57	70.39	76.88
Census income	77.08	83.93	84.29	77.08	83.93	84.29
Thyroid	91.09	92.57	93.472	91.09	92.568	93.473

Table 11 Comparison of fidelity (in %)

Datasets	80–20 partition			70–30 partition		
	Re-RX	RxREN	ERENNR	Re-RX	RxREN	ERENNR
Credit approval dataset	73.28	95.42	97.71	89.29	91.33	95.41
Australian credit approval	66.67	72.46	94.2	76.33	96.14	85.99
Echocardiogram	100	100	91.67	100	100	94.44
Statlog (heart) dataset	72.22	68.52	75.93	60.49	70.37	77.78
Breast cancer	91.18	92.65	97.79	93.63	95.09	97.06
Blood transfusion	99.33	99.33	99.33	95.11	99.11	98.22
German	72.5	69	75	69.33	72.67	80.67
Eye	86.84	76.4	88.85	56.05	59.75	97.37
Pima Indians diabetes	76.62	66.88	86.36	75.65	65.65	82.17
Census income	91.29	95.49	96.04	91.33	95.84	96.31
Thyroid	89.51	90.49	90.14	96.16	96.44	97.27

model desires higher accuracy, precision, specificity, recall, f-measure, MCC and lower FP rate. Good model means that it should have low FP and low FN. But it is always not possible to obtain both FP and FN lower for all cases. In some cases, low FP and high FN, or the reverse may be obtained. In such cases, if maximum of the performance measures are good then the classification model can be considered as a good classifier. But a model having both high FP and high FN cannot be considered as a good classifier.

Table 12 Comparison of fidelity with cross-validation results

Datasets	5-fold cross-validation (in %)			Tenfold cross-validation (in %)		
	Re-RX	RxREN	ERENNR	Re-RX	RxREN	ERENNR
Credit approval dataset	74.35	93.74	95.42	86.92	95.08	96.62
Australian credit approval	68.7	73.91	91.74	73.48	74.78	93.19
Echocardiogram	88.33	91.67	90	91.67	91.67	88.33
Statlog (heart) dataset	72.22	73.70	75.56	72.96	72.22	85.19
Breast cancer	90.59	90.88	94.85	90.44	93.82	96.91
Blood transfusion	98.8	98.93	98.53	95.6	98.13	98
German	69.1	74.1	78	56.67	69.11	69.56
Eye	76.63	73.7	77.5	81.9760	80.9479	77.9039
Pima Indians diabetes	76.36	67.66	85.45	77.92	67.67	86.75
Census income	91.04	93.94	94.26	86.09	94.25	94.42
Thyroid	91.07	91.39	92	90.14	90.31	90.82

Table 13 Confusion matrix for a binary classifier

	Predicted	
	Positive	Negative
Actual		
Positive	TP	FN
Negative	FP	TN

Table 14 Performance measures

Performance measures	Formula
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Recall	$\frac{TP}{TP+FN}$
FP rate	$\frac{FP}{FP+TN}$
Specificity	$\frac{TN}{FP+TN}$
Precision	$\frac{TP}{TP+FP}$
f-measure	$\frac{2*precision*recall}{precision+recall}$
Matthews correlation coefficient (MCC)	$\frac{TP\times TN-FP\times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$

Comparison with Re-RX:

For all the datasets, ERENNR produced less FP and less FN compared to Re-RX except Eye, German, Pima Indians Diabetes, and Census Income datasets. In case of Eye, German, and Census Income datasets ERENNR produced less FP and high FN than Re-RX. In case of Pima Indians Diabetes dataset, ERENNR produced high FP and less FN than Re-RX. But maximum of the performance measures are better for

Table 15 Performance of ERENNR, Re-RX and RxREN algorithm on Credit Approval dataset

Model	Accuracy (%)	Precision (%)	FP rate (%)	Specificity (%)	Recall (%)	*f* measure (%)	MCC (%)
ERENNR	85.20	75.47	23.01	76.99	96.39	84.66	72.75
Re-RX	81.12	72.55	24.78	75.22	89.16	80.00	63.67
RxREN	81.12	70.18	30.09	69.91	96.39	81.22	66.41

Table 16 Performance of ERENNR, Re-RX and RxREN algorithm on Australian Credit Approval dataset

Model	Accuracy (%)	Precision (%)	FP rate (%)	Specificity (%)	Recall (%)	*f* measure (%)	MCC (%)
ERENNR	85.99	92.00	5.22	94.78	75.00	82.63	72.14
Re-RX	75.36	73.03	20.87	79.13	70.65	71.82	49.97
RxREN	84.54	78.30	20.00	80.00	90.22	83.84	69.80

Table 17 Performance of ERENNR, Re-RX and RxREN algorithm on Echocardiogram dataset

Model	Accuracy (%)	Precision (%)	FP rate (%)	Specificity (%)	Recall (%)	*f* measure (%)	MCC (%)
ERENNR	100	100	0	100	100	100	100
Re-RX	94.44	75.00	6.67	93.33	100	85.71	83.67
RxREN	94.44	75.00	6.67	93.33	100	85.71	83.67

Table 18 Performance of ERENNR, Re-RX and RxREN algorithm on Statlog (heart) dataset

Model	Accuracy (%)	Precision (%)	FP rate (%)	Specificity (%)	Recall (%)	*f* measure (%)	MCC (%)
ERENNR	77.78	77.36	35.29	64.71	87.23	82.00	53.90
Re-RX	67.90	59.52	36.17	63.83	73.53	65.79	36.90
RxREN	72.84	73.58	41.18	58.82	82.98	78.00	43.38

Table 19 Performance of ERENNR, Re-RX and RxREN algorithm on breast cancer dataset

Model	Accuracy (%)	Precision (%)	FP rate (%)	Specificity (%)	Recall (%)	*f* measure (%)	MCC (%)
ERENNR	97.06	92.59	2.63	97.37	96.15	94.34	92.38
Re-RX	93.14	81.67	7.24	92.76	94.23	87.50	83.21
RxREN	95.59	92.16	2.63	97.37	90.38	91.26	88.32

Table 20 Performance of ERENNR, Re-RX and RxREN algorithm on blood transfusion dataset

Model	Accuracy (%)	Precision (%)	FP rate (%)	Specificity (%)	Recall (%)	*f* measure (%)	MCC (%)
ERENNR	75.56	100	0	100	1.79	3.51	11.61
Re-RX	74.67	12.50	4.14	95.86	1.79	3.13	– 5.50
RxREN	75.56	100	0	100	1.79	3.51	11.61

Table 21 Performance of ERENNR, Re-RX and RxREN algorithm on German dataset

Model	Accuracy (%)	Precision (%)	FP rate (%)	Specificity (%)	Recall (%)	*f* measure (%)	MCC (%)
ERENNR	73	80.29	44.09	55.91	80.68	80.48	36.70
Re-RX	71	75.90	71.43	28.57	87.50	81.29	19.21
RxREN	67	70.61	82.80	17.20	89.37	78.89	9.14

Table 22 Performance of ERENNR, Re-RX and RxREN algorithm on eye dataset

Model	Accuracy (%)	Precision (%)	FP rate (%)	Specificity (%)	Recall (%)	*f* measure (%)	MCC (%)
ERENNR	65.55	66.67	0.48	99.52	1.79	3.49	6.51
Re-RX	55.03	40.89	50.78	49.22	65.94	50.48	14.54
RxREN	65.26	50.03	32.44	67.56	60.95	54.95	27.48

Table 23 Performance of ERENNR, Re-RX and RxREN algorithm on Pima Indians diabetes dataset

Model	Accuracy (%)	Precision (%)	FP rate (%)	Specificity (%)	Recall (%)	*f* measure (%)	MCC (%)
ERENNR	79.13	94.59	1.34	98.66	43.21	59.32	54.43
Re-RX	69.13	100	0	100	12.35	21.98	28.92
RxREN	71.30	56.88	31.54	68.46	76.54	65.26	43.05

Table 24 Performance of ERENNR, Re-RX and RxREN algorithm on census income dataset

Model	Accuracy (%)	Precision (%)	FP rate (%)	Specificity (%)	Recall (%)	*f* measure (%)	MCC (%)
ERENNR	76.72	100	0	100	1.96	3.84	12.24
Re-RX	70.35	23	8.84	91.16	8.1	11.98	– 1.13
RxREN	76.18	17.65	0.14	99.86	0.093	0.19	– 0.51

Table 25 Performance of ERENNR, Re-RX and RxREN algorithm on thyroid dataset

Model	Accuracy (%)	Precision (%)	FP rate (%)	Specificity (%)	Recall (%)	*f* measure (%)	MCC (%)
ERENNR	95.31	92.96	3.52	96.48	92.96	92.96	89.44
Re-RX	93.95	90.93	4.54	95.46	90.93	90.93	86.39
RxREN	94.94	92.41	3.80	96.2	92.41	92.41	88.61

ERENNR. So, it can be said that the performance of ERENNR is better than Re-RX for those datasets.

Comparison with RxREN:

ERENNR have shown better performance than RxREN for Credit Approval, Echocardiogram, Statlog (Heart), Census Income, and Thyroid datasets. ERENNR produced less FP and less FN for those datasets. For Blood Transfusion dataset performance for ERENNR is equal to RxREN for the selected fold. For Breast Cancer dataset, all the performance measures are equal or better because ERENNR produced equal FP and less FN compared to RxREN.

For German, Australian Credit Approval, Eye, and Pima Indians Diabetes datasets, all the performance measures are better except one or two measures because ERENNR produced less FP and high FN. But maximum of the performance measures is better for ERENNR, so it can be said that the performance of ERENNR is better than RxREN for those datasets.

The algorithm is also validated with a suitable non- parametric statistical test for multiple comparisons: Freidman test, followed by a post hoc test: Least Significant Difference (LSD). Goal of any hypothesis test is to check whether the null hypothesis is rejected or not, and it is decided by the P value. If the P value is very small, then the null hypothesis is rejected.

Null hypothesis for this test:

“There is no significant difference between Re-RX, RxREN, and ERENNR”

Tenfold cross-validation results are used for the test. Table 26 shows the results for Friedman test. Table shows mean value for each model, Freidman statistics: Chi square value and P value. In all the cases except echocardiogram and blood transfusion datasets, the null hypothesis is rejected at a significant ($P<0.05$) or highly significant ($P<0.01$) P value.

Friedman test indicated that there is a significant difference between Re-RX, RxREN and ERENNR algorithms, but it is not clear that which specific algorithm is the reason behind this difference. So, LSD post hoc analysis is performed after Freidman test. LSD performs all possible pairwise comparison of group means obtained from Freidman test. Table 27 shows the results for LSD test. P value for pairs: (ERENNR and Re-RX) and (ERENNR and RxREN) is significant ($P<0.05$)

or highly significant ($P < 0.01$) in all datasets except echocardiogram and blood transfusion datasets. Analyzing all the results in Tables 26 and 27, it can be said that ERENNR algorithm is the reason for significant difference between the three algorithms. So, ERENNR algorithm is statistically significant compared to Re-RX and RxREN algorithms.

The statistical significance of ERENNR algorithm over Re-RX algorithm and RxREN algorithm is also validated based on results from all the 11 datasets together. The mean testing accuracies of tenfolds for each of the datasets by the three algorithms are used for this purpose. Tables 28 and 29 show the results for Friedman test and LSD post hoc test, respectively, over all the 11 datasets. Results for Friedman test show that the mean rank for ERENNR is higher than Re-RX and RxREN, and the null hypothesis is rejected at a highly significant ($P < 0.01$) P value. LSD test further shows that ERENNR algorithm is significant than Re-RX algorithm at a confidence level of 99%, i.e., $P < 0.01$ and ERENNR algorithm is significant than RxREN algorithm at a confidence level of 90%, i.e., $P < 0.1$.

Comparison of ERENNR with Some Other Techniques

Table 30 shows the comparison of ERENNR with some other techniques: a standard rule inducing technique (Decision tree), Decision tree created using predicted outputs of NN, and Continuous Re-RX [10, 12]. DT means decision tree and DT_using_NN means decision tree using the predicted outputs of NN. Table shows that ERENNR performed better than all the techniques in all datasets except Australian Credit Approval and Thyroid. However, ERENNR performed better than continuous Re-RX algorithm in both the cases. It also performed better than DT_using_NN for Thyroid dataset.

Discussion

The preceding subsections present the results produced by ERENNR algorithm. Experiments are performed on 11 datasets taken from UCI and keel repository and comparison is done with Re-RX, RxREN, continuous Re-RX, and decision tree algorithms. All the results shown above convey that ERENNR algorithm is able to generate simple rules with good accuracy. The recursive nature of the algorithm enables it to select appropriate data ranges of input attributes which further allow it to generate accurate rule. Friedman test and LSD post hoc analysis are also performed to validate the statistical significance of the proposed ERENNR. Other than accuracy, rules can be evaluated based on some other measures like:

Table 26 Multiple comparisons using Friedman statistical test

Datasets	Models	Mean ranks	Chi-sq.	*P* value
Credit approval dataset	Re-RX	1.45	9.0556	0.0108*
	RxREN	1.85		
	ERENNR	**2.7**		
Australian credit approval	Re-RX	1.4	14.973	0.0005**
	RxREN	1.65		
	ERENNR	**2.95**		
Echocardiogram	Re-RX	1.9	4.0000	0.1353
	RxREN	1.9		
	ERENNR	**2.2**		
Statlog (heart) dataset	Re-RX	1.4	10.2162	0.006**
	RxREN	1.85		
	ERENNR	**2.75**		
Breast cancer	Re-RX	1.35	13.1667	0.001**
	RxREN	1.8		
	ERENNR	**2.85**		
Blood transfusion	Re-RX	1.75	3.8	0.1496
	RxREN	2.1		
	ERENNR	**2.15**		
German	Re-RX	1.85	8.359	0.0153*
	RxREN	1.45		
	ERENNR	**2.7**		
Eye	Re-RX	1.2	15.3684	0.00046**
	RxREN	1.9		
	ERENNR	**2.9**		
Pima Indians diabetes	Re-RX	1.4	10.974	0.0042**
	RxREN	1.8		
	ERENNR	**2.8**		
Census income	Re-RX	1	20	0.000045**
	RxREN	2		
	ERENNR	**3**		
Thyroid	Re-RX	1	20	0.000045**
	RxREN	2		
	ERENNR	**3**		

Bold values in the mean rank column shows the highest mean ranks for the respective datasets

*Significant at $P < 0.05$

**Highly significant at $P < 0.01$

Fidelity:

Fidelity is the measure of ability of rules to mimic the behavior of NN. ERENNR produced rules with good fidelity (shown in Table 11 and 12).

Table 27 Pairwise comparisons using least significant difference (LSD) post hoc test

Datasets	Models	Models	*P* value
Credit approval dataset	Re-RX	RxREN	0.345778
	Re-RX	ERENNR	0.003216**
	RxREN	ERENNR	0.045126*
Australian Credit approval	Re-RX	RxREN	0.561079
	Re-RX	ERENNR	0.000313**
	RxREN	ERENNR	0.002507**
Echocardiogram	Re-RX	RxREN	1
	Re-RX	ERENNR	0.083264
	RxREN	ERENNR	0.083264
Statlog (heart) dataset	Re-RX	RxREN	0.295455
	Re-RX	ERENNR	0.001697**
	RxREN	ERENNR	0.036397*
Breast cancer	Re-RX	RxREN	0.288844
	Re-RX	ERENNR	0.000406**
	RxREN	ERENNR	0.013328*
Blood transfusion	Re-RX	RxREN	0.117524
	Re-RX	ERENNR	0.073638
	RxREN	ERENNR	0.823063
German	Re-RX	RxREN	0.502334
	Re-RX	ERENNR	0.049*
	RxREN	ERENNR	0.0046**
Eye	Re-RX	RxREN	0.108293
	Re-RX	ERENNR	0.000096**
	RxREN	ERENNR	0.021781*
Pima Indians diabetes	Re-RX	RxREN	0.358795
	Re-RX	ERENNR	0.001318**
	RxREN	ERENNR	0.021781*
Census income	Re-RX	RxREN	0.025347*
	Re-RX	ERENNR	0.000007**
	RxREN	ERENNR	0.025347*
Thyroid	Re-RX	RxREN	0.0253*
	Re-RX	ERENNR	0.00000774**
	RxREN	ERENNR	0.0253*

*Significant at $P < 0.05$

**Highly significant at $P < 0.01$

Table 28 Multiple comparisons over multiple datasets using Friedman statistical test

Models	Mean Ranks	Chi-sq.	*P* value
Re-RX	1.3462	12.0417	0.0024
RxREN	2		
ERENNR	2.6538		

Table 29 Pairwise comparisons using least significant difference (LSD) post hoc test over multiple datasets

Models	Models	*P* value
Re-RX	RxREN	0.0827
Re-RX	ERENNR	0.00052024
RxREN	ERENNR	0.0827

Table 30 Accuracy comparison of ERENNR and decision tree with tenfolds cross-validation results (in %)

Datasets	DT	DT_using_NN	Continuous_ ReRX	ERNNNR
Credit approval dataset	85.85	85.69	84.15	86.62
Australian credit approval	86.96	85.94	84.20	85.51
Echocardiogram	88.33	83.33	95	96.67
Statlog (heart) dataset	75.19	75.16	74.07	80.37
Breast cancer	93.68	95	94.26	96.47
Blood transfusion	74.8	74.27	71.3	76.54
German	72.5	71	69.4	73.1
Eye	54.03	55.15	54.03	55.24
Pima Indian diabetes	75.58	73.38	73.77	76.88
Census	83.7	83.26	80.89	84.14
Thyroid	96.9	91.56	92.58	93.47

Scalability:

ERENNR is scalable in the sense that pruning and rule extraction steps are independent. This rule extraction technique can be used for neural network that is pruned with any other techniques.

Portability:

Though it extracts knowledge from each node of the network, it does not analyze the nodes based on the weights. It uses pedagogical approach at each node which extracts knowledge only in the form of input to the node and output of the node. So, it can be used for neural network with any structure.

Comprehensibility:

In terms of comprehensibility, due to recursive nature, the algorithm may produce less comprehensible (global comprehensibility) rules than some pedagogical rule extraction algorithm, but its local comprehensibility (number of conditions in a rule) is better than some existing decompositional and eclectic rule extraction algorithms which use decision tree to extract rules.

Conclusion

This paper proposes an eclectic rule extraction algorithm which extracts rules recursively from a single hidden layer neural network. The algorithm analyzes each node and extracts knowledge from each node in the form of input to the node and output from the node. And, finally by combining knowledge obtained from each of the nodes, it generates global rule set in the form of input data ranges and outputs. Subspace of each rule in the rule set is subdivided, if certain conditions are satisfied. To validate the algorithm, eleven datasets are taken from UCI and keel repository. The whole algorithm is illustrated with a dataset in the illustrative example section.

ERENNR is compared with Re-RX, RxREN, continuous Re-RX and decision tree algorithms. Results show that ERENNR is able to generate simple rules with good accuracy and fidelity. The performance of ERENNR is also validated based on different performance measures calculated using confusion matrix and hypothesis testing. So, in a nutshell, the algorithm can be said to be effective.

The algorithm extracts rules from neural networks with single hidden layer. In future, the work may be extended to multi-hidden layers.

References

1. Anbananthen, S.K., Sainarayanan, G., Chekima, A., Teo, J.: Data mining using pruned artificial neural network tree (ANNT). Inf. Commun. Technol. **1**, 1350–1356 (2006)
2. Augasta, M.G., Kathirvalavakumar, T.: Rule extraction from neural networks—a comparative study. International Conference on Pattern Recognition. Informatics and Medical Engineering, Salem, Tamilnadu, 404–408 (2012)
3. Augusta, M.G., Kathirvalavakumar, T.: Reverse engineering the neural networks for rule extraction in classification problems. Neural Process. Lett. **35**(2), 131–150 (2012)
4. Biswas, S.K., Chakraborty, M., Singh, H.R., Devi, D., Purkayastha, B., Das, A.K.: Hybrid case-based reasoning system by cost-sensitive neural network for classification. Soft Comput. **24**, 1–18 (2016)
5. Biswas, S.K., Chakraborty, M., Purkayastha, B., Thounaojam, D.M., Roy, P.: Rule extraction from training data using neural network. Int. J. Artif. Intell. Tool **26**, 3 (2017)
6. Chakraborty, M., Biswas, S.K., Purkayastha, B.: Recursive rule extraction from NN using reverse engineering technique. New Gener. Comput. **36**(2), 119–142 (2018)
7. Craven, M., Shavlik, J.: Extracting tree-structured representations of trained network. *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge **8**, 24–30 (1996)
8. Etchells, T.A., Lisboa, P.J.G.: Orthogonal search-based rule extraction (OSRE) for trained neural networks: a practical and efficient approach. IEEE Trans. Neural Networks **17**(2), 374–384 (2006)
9. Fortuny, E.J., Martens, D.: Active learning-based pedagogical rule extraction. IEEE Trans. Neural Netw. Learn. Syst. **26**(11), 2664–2677 (2015)
10. Hayashi, Y., Nakano, S., Fujisawa, S.: Use of the recursive-rule extraction algorithm with continuous attributes to improve diagnostic accuracy in thyroid disease. Inf. Med. Unlocked **1**, 1–8 (2015)
11. Hayashi, Y., Yukita, S.: Rule extraction using recursive-rule extraction algorithm with J48graft combined with sampling selection techniques for the diagnosis of type 2 diabetes mellitus in the Pima Indian dataset. Inf. Med. Unlocked **2**, 92–104 (2016)
12. Hayashi, Y.: Application of a rule extraction algorithm family based on the Re-RX algorithm to financial credit risk assessment from a Pareto optimal perspective. Oper. Res. Perspect. **3**, 32–42 (2016)

13. Hailesilassie, T.: Ensemble neural network rule extraction using Re-RX algorithm. Int. J. Comput. Sci. Inf. Sec. **14**, 7 (2016)
14. Hara, A., Hayashi, Y.: Ensemble neural network rule extraction using re-RX algorithm. Neural Networks (IJCNN), 1–6 (2012)
15. Iqbal, R.A.: Eclectic rule extraction from neural networks using aggregated decision trees. IEEE, 7th International Conference on Electrical & Computer Engineering (ICECE), 129–132 (2012)
16. Jivani, K., Ambasana, J., Kanani, S.: A survey on rule extraction approaches based techniques for data classification using neural network. Int. J. Futuristic Trends Eng. Technol. **1**(1), 4–7 (2014)
17. Kumar, S.: Neural networks: a classroom approach, 2nd edn. Tata McGraw-Hill Education, New Delhi (2004)
18. Odajimaa, K., Hayashi, Y., Tianxia, G., Setiono, R.: Greedy rule generation from discrete data and its use in neural network rule extraction. Neural Netw. **21**(7), 1020–1028 (2008)
19. Permanasari, A. E., Rambli, D.R.A., Dominic, P.D.D.: Forecasting of salmonellosis incidence in human using Artificial Neural Network (ANN). Computer and Automation Engineering (ICCAE), The 2nd International Conference, 1, 136–139 (2010)
20. Sestito, S., Dillon, T.: Automated knowledge acquisition of rules with continuously valued attributes. In: Proceedings of 12th International Conference on Expert Systems and their Applications, 645–656 (1992)
21. Setiono, R., Liu, H.: Symbolic representation of neural networks. IEEE Comput. **29**(3), 71–77 (1996)
22. Setiono, R.: Extracting rules from neural networks by pruning and hidden-unit splitting. Neural Comput. **9**(1), 205–225 (1997)
23. Setiono, R., Liu, H.: NeuroLinear: from neural networks to oblique decision rules. Neurocomputing **17**(1), 1–24 (1997)
24. Setiono, R., Kheng, W.: FERNN: an algorithm for fast extraction of rules from neural networks. Appl. Intell. **12**(1), 15–25 (2000)
25. Setiono, R., Baesens, B., Mues, C.: Recursive neural network rule extraction for data with mixed attributes. IEEE Trans. Neural Networks **19**(2), 299–307 (2008)
26. Taha, I.A., Ghosh, J.: Symbolic interpretation of artificial neural networks. IEEE Trans. Knowl. Data Eng. **11**(3), 448–463 (1999)
27. Towel, G., Shavlik, J.: The extraction of refined rules from knowledge based neural networks. Mach. Learn. **13**(1), 71–101 (1993)
28. Zhou, Z.H.: Rule extraction: using neural networks or for neural networks? J. Comput. Sci. Technol. **19**(2), 249–253 (2004)